2.12 ATMOSPHERIC CHARACTERISTICS

For mission-level combat models, the environment is defined to consist of the natural and man-made atmospheric, topographic, and bathymetric conditions in which combat occurs. Environmental phenomena are usually considered to be those which can affect most or all of the scenario without being directly attributable to a single entity within the scenario.

The environment can produce many types of effects in a mission-level combat scenario. Movement can be affected by several aspects of the environment: surface topography can affect the movement of tanks, mobile SAMs, etc.; the atmosphere can affect aircraft movement; and the ocean surface or subsurface can affect movement of ships and submarines. Other types of physical or psychological effects could be caused by time of day and time of year, sunspots, the earth's magnetic field, ephemeris data, etc. However, the effects considered in this functional area template are the environmental effects on energy transmission.

2.12.1 Functional Element Design Requirements

The design requirements for the atmospheric characteristics functional element are:

a. The model will provide the capability to represent energy losses and/or gains due to the transmission medium according to user instructions. The user may define loss/gain values that depend on frequency, the altitude of the transmitter, the altitude of the receiver, and the ground distance between the transmitter and the receiver. If the user does not define these loss/gain values, only the standard 1/R2 losses due to distance are included in the received energy calculations. These loss/gain values will be applied to energy transmitted for sensing, communicating, and jamming.

b. The model will represent the effects of electromagnetic refraction in the lower atmosphere by allowing the user to specify the effective radius of the earth. In scenarios where refraction is used, the model will check for line of sight using this effective radius rather than the true radius.

c. The model will provide a option to select atmospheric conditions corresponding to either day or night.

2.12.2 Functional Element Design Approach

Design Element 12-1: Attenuation

In Suppressor, attenuation is modeled by associating a TRANSMISSION-LOSS table with a sensor receiver. This table defines loss values in dB with dimensions for frequency, transmitter altitude, receiver altitude, and 2d-distance. The table is a Suppressor irregular interval table allowing the user to choose the resolution of the table with the dimensions entered in the table. The code performs one table look-up for infrared, optical, or warning receiver sensors, as well as jammer or communications systems. If a radar sensor has an associated TRANSMISSION-LOSS table, then one table look-up is performed to determine the loss for the energy radiated from the transmitter out to the target, and then a second table look-up is performed to determine the loss of the energy reflected from the

target back to the radar receiver. If no table is associated with a receiver, then the signal is not attenuated.

Design Element 12-2: Refraction/Ducting

Refraction and ducting refer to perturbations in the energy used to sense a target as the energy travels through the atmosphere. Suppressor includes models for four distinct types of sensors (radar, infrared, optical, and warning receiver). Each sensor type includes a specific sensing equation used to determine the ability of the sensor to detect the target. Each of these sensing equations include terms to represent generic losses as the energy travels through the atmosphere. This functional element description focuses on the controlling module for sense events. For a detailed explanation of each individual type of sensing equation, refer to the 2.0 Sensors functional element descriptions for the Platform Aircraft / SAM / TBM template.

Four distinct types of sensor models are included in Suppressor: radar, infrared, optical, and warning receiver. Each sensor type includes a specific sensing equation used to determine the ability of the sensor to detect the target. Each of these sensing equations includes terms to represent generic losses as the energy travels through the atmosphere, and these terms are defined by analysts in the TDB input file.

Design Element 12-3: Time of Day

This functional element is used to distinguish day versus night in the simulation of a scenario. Suppressor includes a day/night flag which is defined in the input, and the day/night status can be used in a player's tactics to influence decisions.

Suppressor analysts can define RESOURCE-ALLOCATION tactics in which the decisions differ depending on whether it is day or night. The criterion is called SKY-RADIANCE, and its value for a simulation is defined following the SKY RADIANCE: phrase in the model input file. SKY-RADIANCE can be used for any tactical decisions controlled through RESOUREC-ALLOCATION, for example SKY-RADIANCE can be used as a criterion for turning optical sensors on during day and off during night.

2.12.3 Functional Element Software Design

Attenuation Module Design

Subroutine ERGATN is a utility module which determines the transmission loss term from a specific table with frequency, altitude of the transmitter (source of the signal), altitude of the receiver, and distance as input arguments.

The design for Subroutine ERGATN follows:

```
*begin logic to determine transmission losses:
   *when transmission loss table exists:
      *perform lookups to determine atmospheric attenuation;
      *find table interval for frequency dimension;
      *find table interval for transmitter altitude dimension;
      *find table interval for receiver altitude dimension;
      *find table interval for 2d-distance dimension;
      *get transmission-loss term from table;
   *end of test for transmission loss table exists.
*end of logic for ERGATN.
```

Refraction/Ducting Module Design

Subroutine OBSCHA is the controlling module for every type of sense event. It is used to perform several preparation steps, and then invoke a module for the specific type of sensor being modeled. This is the top-level design for Subroutine OBSCHA:

```
ABSTRACT     supervise sensor chance calculations
   *begin logic to supervise sensor chance calculations:
      *invoke logic to get position of target;
      *invoke logic to get receiver position;
      *invoke logic to get transmitter position if needed;
      *invoke logic to get length of group list under target;
      *allocate perceived group list;
      *invoke logic to get projected cross section of target;
      *when group list has been revised:
         *allocate revised group list;
         *copy data into revised list and recycle original
      *end of test for revised group list.
      *look up old status flags and initialize new flags to old;
      *calculate 3-D range from sensor to target;
      *initialize various sensing flags;
      *when minimum or maximum doppler speed limits defined:
         *calculate relative speed of target along range
         *when below defined minimum doppler level:
            *set flag to not enough doppler
         *otherwise, above minimum or minimum not defined:
            *set flag if doppler above defined maximum
         *end of test for doppler tests.
         *set new detection flags for doppler results
      *end of test for doppler limits defined.
      *look up sensor type (radar,eo,ir,rwr);
      *when doppler limits not violated:
         *when this sensor is a warning receiver:
            *loop, while target elements left to check:
               *loop, while more systems in this element:
                  *when system can be intercepted by the rwr:
                     *determine target transmitter vertical offset;
                  *end of test if system can be intercepted.
               *end loop, while more systems in this element.
            *end loop, while target elements left to check.
         *end of test for a warning receiver.
         *invoke logic to determine within sensing limits;
         *when tracking this target:
            *look up post lockon value if enough elapsed time;
         *end of test for tracking this target.
         *when sensor is external:
            *extract external name;
            *call external processing routine;
         *otherwise, regular sensor:
         *when radar sensor:
            *invoke logic to perform radar sensor calculations;
         *but, when optical sensor:
            *invoke logic to perform optical sensor calculations;
         *but, when infrared sensor:
            *invoke logic to perform infrared sensor calculations;
         *but, when warning receiver sensor:
            *invoke logic to perform rwr sensor calculations;
         *end of test for sensor type, within sensing limits.
         *end of test for external sensor.
         *set signal-to-noise flag in detection status;
      *end of test for doppler limits not violated.
      *allocate storage and store flags
      *when target was not in range:
```

```
         *when target is now in range:
            *change flag to has been in range;
            *write out "sensor in range (111) incident;
         *end of test for now in range.
      *end of test for not in range before.
      *when old and new status codes are different:
         *allocate storage and store flags;
         *invoke logic to write out situation;
         *determine success(1) or failure(0) from old status code;
         *determine success(1) or failure(0) from new status code;
         *when there is a change in overall status:
            *write "change-in-sensing-status-for" incident;
         *end of test for change in overall status.
      *end of test for old and new status codes different.
      *write "now-sensing-elements" output incident;
      *store new status flags
      *invoke logic to prepare sensor results report;
      *when any reportable results and tracking this target:
         *when countermeasure table:
            *loop, while jammers left that must be checked:
               *when implicit detail and jammer is on:
                  *when jammer belongs to target cluster:
                     *set flag if implicit interaction;
                  *end of test for jammer, target in same cluster.
               *end of test for implicit detail and jammer on.
            *end of loop for jammers.
         *end of test for countermeasure table.
         *store result if random effective countermeasures;
         *invoke logic to determine tracking results;
      *end of test for reportable results and tracking this target.
   *end of logic for OBSCHA.
```

Radiance/Transmittance Module Design

Subroutine OBSCHA is the primary module for all Suppressor sense events. The software design for OBSCHA is given in the preceding functional element, 12-2 Refraction/ Ducting.

Module OBSRDR is invoked from Subroutine OBSCHA whenever the sensor is a radar system. This is the top-level design for Subroutine OBSCHA:

```
   *begin logic to perform radar sensor calculations:
      *when within sensing limits:
         *look up time of sense chance;
         *look up target position pointer;
         *look up receiver position pointer;
         *look up receiver data pointers;
         *calculate range from sensor to target;
         *invoke logic to calculate jammer signals;
         *calculate square of range from xmtr to target;
         *calculate range to target;
         *look up peak transmitter power out;
         *adjust power for pulse doppler, pulse compression
         *invoke logic to target signal calculations;
         *when clutter is modeled:
            *perform lookup to determine clutter power;
            *when terrain data:
               *invoke logic to get terrain altitude under target;
            *end of test for terrain.
         *end of test for clutter.
         *look up internal receiver noise;
         *evaluate visibility for target vs clutter plus noise;
```

```
         *calculate total noise level;
         *when pulse jamming is present:
            *set interference to pulse jamming level;
            *use pulse jamming detection threshold;
         *but, when noise predominates:
            *set interference to total noise level;
            *determine threshold based on noise jamming level;
         *end of test for jamming modulation.
         *evaluate visibility for target vs interference;
         *when target is visible over clutter+noise and jamming:
            *target is visible;
            *when operational-probability-of-detection available:
               *determine current radar operator workload;
                  *add other targets being tracked to workload;
               *find interval in prob-detect data for workload;
               *find interval in prob-detect data for s/n level;
               *find perception block for target, if it exists;
               *when current target has an established track:
                  *use cued probability of detection;
               *but, when current target has external source:
                  *use external-cue probability of detection;
               *otherwise current target is new:
                  *use uncued probability of detection;
               *end of type of cue determination.
               *when draw is above the probability-of-detection:
                  *set result to failed sense event, clear jam flag;
               *end of test for draw above probability-of-detection.
            *end of test for operational-probability-of-detection.
         *end of test for target visibility.
         *when jammer is visible over internal noise:
            *jamming is present;
         *end of test for target visibility and jammer effectiveness.
         *when operator thinks the radar is being jammed:
            *set operator thinks the radar is being jammed flag;
            *when there are alternate frequencies:
               *when not currently changing frequency:
                  *get next available frequency;
                  *schedule change frequency event;
                  *write out "starts to change freq" message;
               *end of test if not currently changing frequency.
            *end of test for alternate frequencies.
         *otherwise, operator thinks the radar is not jammed:
            *clear operator thinks the radar is being jammed flag;
         *end of test if operator thinks the radar is jammed.
         *store flag and time operator thinks the radar is jammed;
         *when jamming interfered:
            *write is-jammed-(explicitly) message;
         *end of test if jamming interfered.
      *end of test for within sensing limits.
   *end of logic for OBSRDR.
```

Module OBSOPT is invoked from Subroutine OBSCHA whenever the sensor is an optical system. This is the top-level design for Subroutine OBSOPT:

```
   *begin logic to perform optical sensor calculations:
      *look up pointer to sensor/target buffer, receiver data;
      *when within sensing limits:
         *look up target, sensor position pointers;
         *look up receiver data pointers;
         *calculate range from sensor to target;
         *look up transmission loss;
         *look up background and path radiance;
         *compute background radiance at the sensor;
```

```
         *compute solid angle of target at sensor;
         *when background radiance is zero:
            *loop, for search, reacq, and track glimpse tables:
               *get highest contrast dimension value from tables;
            *end of loop for tables.
         *but, when background radiance not zero:
            *compute target contrast at the sensor;
         *end of test for background radiance.
         *look up single glimpse prob for search, reacq, track;
      *but, when target not within limits:
         *set single glimpse probabilities to zero;
      *end of test for within limits.
      *look up cumulative detection probabilities for target;
      *determine number of glimpses in reacquisition interval;
      *initialize sensing rate to acquisition mode;
      *adjust rate depending on track and fire conditions;
      *compute sensing cycle time;
      *loop, for number of glimpses in sensing interval:
         *calculate probability of search, reacq, track;
      *end of loop for number of glimpses.
      *store cumulative probabilities for next sense chance;
      *evaluate visibility for target;
   *end of logic for OBSOPT.
```

Module OBSIR is invoked from Subroutine OBSCHA whenever the sensor is an infrared system. This is the top-level design for Subroutine OBSIR:

```
   *begin logic to perform infrared sensor calculations:
      *when within sensing limits:
         *look up target, sensor position pointers;
         *look up receiver data pointers;
         *calculate range from sensor to target;
         *look up transmission loss;
         *calculate signal-to-noise ratio;
         *when target angle at sensor exceeds field of view:
            *adjust signal-to-noise ratio;
         *end of test for pixel field of view.
         *evaluate visibility for target vs interference;
      *end of test for within sensing limits.
   *end of logic for OBSIR.
```

Module OBSLSN is invoked from Subroutine OBSCHA whenever the sensor is a warning receiver system. This is the top-level design for Subroutine OBSLSN:

```
   *begin logic for emitter receiver calculations:
      *when target within sensing limits:
         *initialize output pointer and flags;
         *look up antenna pointing info and pattern for receiver;
         *calculate constant part of one-way radio equation;
         *look up or initialize values for candidate groups;
         *initialize upper and lower frequency limits for receiver;
         *look up mainbeam/backlobe detection criteria;
         *loop, while target groups left to be checked:
            *loop, while systems left to be checked:
               *when sensor found on interaction table:
                  *set values used to determine transmission losses;
                  *when system is a comm transmitter:
                     *when the transmitter is operating:
                        *determine the status of the transmitter;
                        *when the transmitter is transmitting:
                           *look up frequency, power, loss, net type;
                           *determine transmission loss;
```

```
                        *set mainbeam option (default)
                     *end of check for continuous transmission mode,
                     * or transmitter is transmitting.
                  *end of check for the transmitter is operating.
               *but, when an emitting sensor transmitter:
                  *look up frequency, power, losses;
                  *determine transmission loss;
                  *determine detection options;
                  *invoke emitter pointing if mainbeam desired;
                  *set default calculations if no criteria chosen;
               *end of test for comm or sensor transmitter.
               *when calculations needed and frequency limits:
                  *check for emissions within limits;
               *end of test for calculations and limits.
               *when calculations should be done:
                  *initialize flags to no detection;
                  *process comm or mainbeam calc if required;
                  *process backlobe calculation if required;
                  *cumulate detection results for target cluster;
                  *when enough signal exists for detection:
                     *allocate emitter results and add to list;
                     *store the sensor type code, sense time, id;
                  *end of test for signal level.
               *end of test for calculations to be performed.
            *end of test for interaction table.
         *end of loop for systems.
         *when internal success flag was set:
            *increment counter and add group id to list;
         *end of test for successful sensing.
      *end of loop for target groups.
      *when any target groups sensed:
         *when group list was revised:
            *allocate new perceived group list and copy the data;
            *recycle temporary perceived group list;
         *end of test for revised group list.
      *otherwise, no target groups seen:
         *store saved number of groups;
      *end of test for target groups sensed.
      *set mainbeam/backlobe results if applicable;
   *end of test for within sensing limits.
*end of logic for OBSLSN.
```

Time of Day Module Design

Subroutine EVALU8 is used in Suppressor to evaluate all of the RESOURCE-ALLOCATION criteria. This is the design for the portion of EVALU8 which copies the current day or night state so it can be used as a factor in the tactical decisions of a player:

```
ABSTRACT     calculate criteria for mental decision events
   *begin to calculate criteria for decisions:
            .
            .
            .
      *loop, while more resource types:
            .
            .
            .
         *loop, while more criteria:
            .
            .
            .
            *but, when sky radiance (54):
               *store sky radiance;
```

```
            .
            .
            .
         *end of loop for more criteria.
      *end of loop for more subordinate types.
   *end of logic for EVALU8.
```

2.12.4 Assumptions and Limitations

Using a loss table associated with each sensor receiver type removes any differences in attenuation that might be due to weather phenomena that are not uniform over the entire region.

The background radiance for infrared or optical sensors detecting a target does not consider the position of the sun. This applies to situations when the sun or other significant factors in target background affect the performance of sensors. Infrared and optical sensors in Suppressor utilize tabular data for background-radiance and solar-irradiance. Both tables include data which vary with target altitude. The orientation of the sun relative to the sensor and target is not currently taken into account by Suppressor.

2.12.5 Known Problems or Anomalies

None.